

# Máster Avanzado de Programación en Python para Hacking, BigData y Machine Learning

Programación Avanzada Python

# LECCIÓN 06

## Paquetes y módulos

# ÍNDICE

Introducción

Objetivos

Programación modular

Búsqueda e instalación de paquetes

Creando nuestros propios módulos

Creando nuestros propios paquetes

# INTRODUCCIÓN

En esta sexta lección vamos a estudiar como trabajar paquetes y módulos. Primero veremos como buscar e instalar diferentes paquetes. Seguidamente veremos de como crear nuestros propios paquetes y módulos con la intención de poder utilizar en otros programas.

# OBJETIVOS

Al finalizar esta lección serás capaz de:

- 1 Los principales tipos de datos de colecciones.
- 2 Trabajar con cadenas, listas, conjuntos y diccionarios.
- 3 Diferenciar cada una de las colecciones.
- 4 Buscar otras funcionales adicionales a estos tipos de datos

## PROGRAMACIÓN MODULAR

La programación modular divide los grandes programas o aplicaciones en un número de pequeñas unidades y los hace reutilizables:

- Los módulos son fáciles de manejar y pueden centrarse fácilmente en la corrección de errores. Si un programa grande se divide en partes más pequeñas, es realmente útil para depurar cada sección.
- Un módulo puede mantener y hacer cambios fácilmente sin conocer todas las partes del programa. Los módulos pueden distinguir sus funciones bajo diferentes nombres para poder ser accedidos fácilmente.
- Una función o atributos bien definidos pueden ser reutilizados en cualquier lugar especificando su nombre y características evitando la redundancia en los programas.

## Módulos

Los módulos son secciones que contienen simplemente una o dos funciones que realizan cualquier tarea. Un módulo puede ser definido como un programa Python y puede importarse en otros programas.

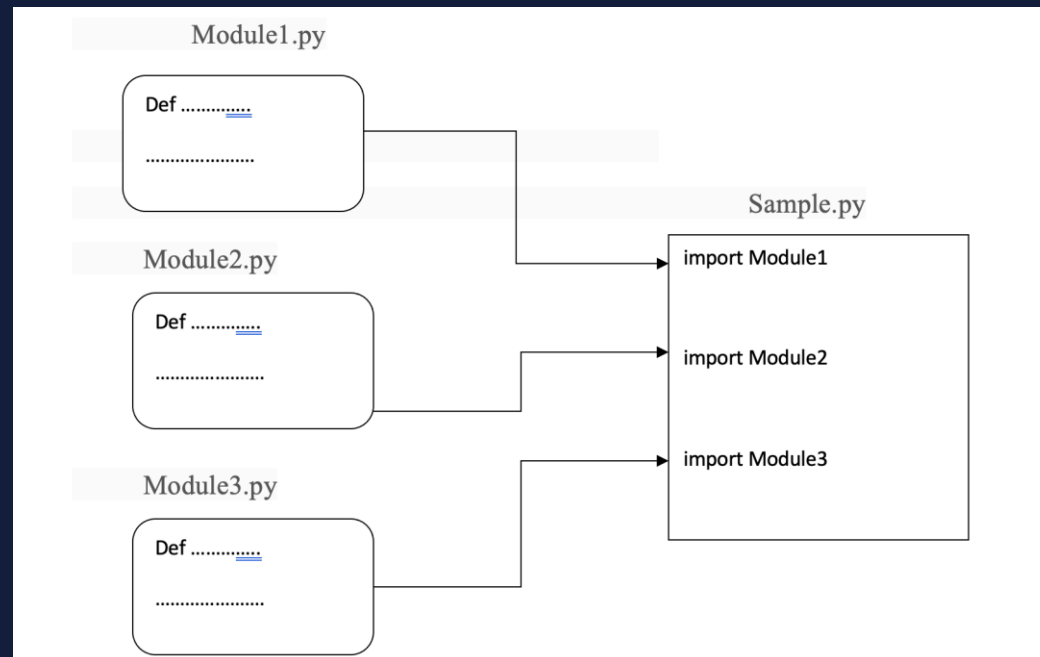
## Paquetes

Los paquetes son una colección de módulos. Estos son ampliamente utilizados en los programas de Python. Los paquetes pueden almacenar muchos módulos y también subpaquetes.



## Import

Un módulo se define como un programa de Python y puede ser accedido a otros programas utilizando la palabra clave **import**.



## BÚSQUEDA E INSTALACIÓN DE PAQUETES

Los paquetes son una colección de módulos. Hay muchos paquetes incorporados en el core de Python que se utilizan ampliamente en los programas que desarrollamos.

**Pip** es un instalador recomendado, que ayuda a instalar y actualizar los paquetes de Python con una sola línea de comando. Para comprobar la versión de pip ejecutamos lo siguiente

## CREANDO NUESTROS PROPIOS MÓDULOS

Un módulo puede ser definido como un programa de Python y se puede importar en otros programas.

```
module.py X
1 def square(n):
2     return n*n
3
4 list = [1,2,3,4]
5 string = "hello"
```

El módulo se guarda en el directorio de trabajo actual y otro programa *sample.py* utiliza las funciones y atributos en el *module.py*. Para ello el programa *sample.py* debe de hacer referencia al nuevo modulo creado:

```
import module
```

## Declaraciones de importación

Los módulos creados son llamados con la sentencia import. Hay diferentes maneras de llamar a los módulos usando import:

- `Import nombre_del_módulo`
- `from nombre_del_módulo import nombre/nombres`
- `from nombre_del_módulo import nombre_alternativo`
- `import nombre_del_módulo como nombre_alternativo`

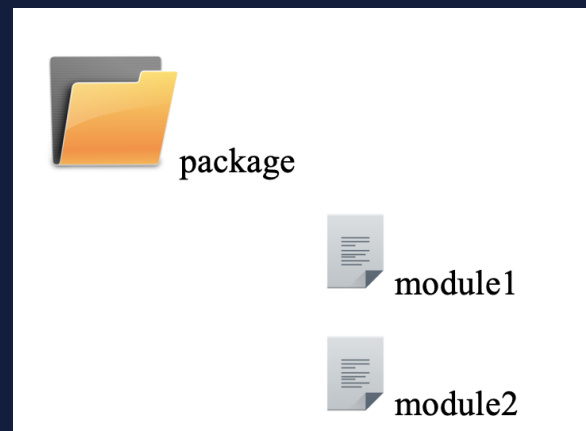
```
from module import square as sqr  
  
print(sqr(2))
```

4

## CREANDO NUESTROS PROPIOS PAQUETES

Cuando se crean muchos módulos, se vuelve más confuso almacenar y acceder en función de su uso. Los paquetes ayudan a agrupar los métodos que tienen la misma funcionalidad juntos.

Los módulos con la misma funcionalidad o para el mismo propósito se pueden guardar bajo un directorio y nombre, el cual será el nombre del paquete.

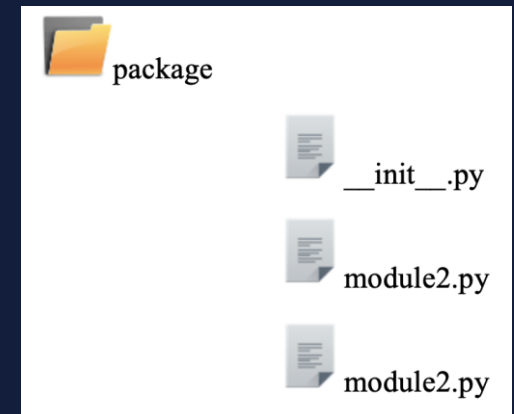




## Inicialización del paquete

Al igual que la función `__init__` dada en la clase es utilizada para la inicialización, aquí el archivo `__init__.py` se utiliza para inicializar los módulos del paquete.

El archivo se define dentro del directorio del paquete al principio, antes de todos los archivos de los módulos. Cuando se importa el paquete, el archivo `init` se invoca automáticamente.



## Sub paquetes

Podemos añadir paquetes dentro del paquete como:

module1.py X

```
1 def mod1():
2 | print("mod1")
```

module2.py X

```
1 def mod2():
2 | print("mod2")
```

module3.py X

```
1 def mod3():
2 | print("mod3")
```

module4.py X

```
1 def mod4():
2 | print("mod4")
```

```
import package.sub_pack1.module1
```

```
package.sub_pack1.module1.mod1()
```

```
mod1
```

```
import package.sub_pack2.module3
```

```
import package.sub_pack2.module4
```

```
package.sub_pack2.module3.mod3()
```


```
package.sub_pack2.module4.mod4()
```


```
mod3
```

```
mod4
```

 package


 sub package1

 module1

 module2

 sub package2

 module3

 module4



## CONCLUSIONES

1

Un módulo se define como un programa de Python y puede ser accedido a otros programas utilizando la palabra clave **import**.

2

**Pip** es un instalador recomendado, que ayuda a instalar y actualizar los paquetes de Python con una sola línea de comando. Para comprobar la versión de pip ejecutamos lo siguiente

3

Los **paquetes** son una colección de módulos. Hay muchos paquetes incorporados en el core de Python que se utilizan ampliamente en los programas que desarrollamos.

MUCHAS GRACIAS POR SU ATENCIÓN



[rsanchezi@grupomainjobs.com](mailto:rsanchezi@grupomainjobs.com)



Rubén Sánchez Iruela

[linkedin.com/in/ruben-sanchez-iruela-8156799a](https://www.linkedin.com/in/ruben-sanchez-iruela-8156799a)



[twitter.com/eiposgrados](https://twitter.com/eiposgrados)



[facebook.com/eiposgrados](https://facebook.com/eiposgrados)



[instagram.com/eiposgrados](https://instagram.com/eiposgrados)